Drone-based Optimal and Heuristic Orienteering Algorithms Towards Bug Detection in Orchards

Francesco Betti Sorbelli*, Federico Corò†, Sajal K. Das†, Lorenzo Palazzetti‡, and Cristina M. Pinotti*
* Department of Computer Science and Mathematics, University of Perugia, Italy
† Department of Computer Science, Missouri Science and Technology University, USA
‡ Department of Computer Science and Mathematics, University of Florence, Italy
Email: *{francesco.bettisorbelli, cristina.pinotti}@unipg.it, †{federico.coro, sdas}@mst.edu, ‡lorenzo.palazzetti@unifi.it

2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS) | 978-1-6654-9512-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/DCOSS54816.2022.00032

***Abstract*—In this paper, we consider the problem of using a drone to collect information within orchards in order to detect bugs. An orchard can be modeled as an aisle-graph, which is a regular data structure formed by consecutive aisles where trees are arranged in a straight line. For monitoring the presence of bugs, a drone flies close to the trees and takes videos and/or pictures that will be analyzed offline. As the drone's energy is limited, only a subset of locations in the orchard can be visited with a fully charged battery. Those places that are most likely to be infested should be selected to promptly detect the parasite. We study the budgeted constrained position selection problem in the orchard from an algorithmic point of view. We present the *Single-drone Orienteering Aisle-graph Problem* (SOAP), a variant of the well-known orienteering problem where the finite resource is the drone's battery. We first show that SOAP can be optimally solved for aisle-graphs in polynomial time. However, the optimal solution is not efficient for large orchards. Then, we propose two efficient heuristics that work even for large (orchard) instances. After a thorough analysis of the proposed solutions, we evaluate their performance by simulation experiments on both synthetic and real data sets.**

I. Introduction

Unmanned aerial vehicles (e.g., drones) are being increasingly used in numerous civilian applications, such as delivery of goods [1], [2], search and rescue operations [3], [4], and precision agriculture [5]. Publications on the use of drones in agriculture started appearing around 1998 and terrifically increased in the last decade, up to more than 150 reports in 2017 [6]. One of the main drivers for the implementation of drone-based remote sensing technologies in agriculture is the potential time saved by automatizing monitoring, making the technology cost-effective for growers [7], [8]. Compared to the conventional platforms for remote sensing, such as ground stations, manned aerial vehicles, or satellites, drones can potentially cover larger areas than ground-based or handheld devices in a shorter time too. They can also fly at much lower altitudes than manned aerial vehicles increasing the spatial resolution of images, and reducing the number of distorted soil and tree pixels [6]. Moreover, drones do not have long revisiting times as satellites. An example of a time-consuming procedure is the traditional field scouting for pest infestations. In such a case, drones can accurately and very frequently inspect the trees by taking high-resolution images of any part (bottom, top) of them. So, combined with context information, drones can discover the pest infection at the initial stage thus reducing the insecticidal or biopesticide consumption [9].

In this paper, we consider an application that uses a drone with video capabilities inside an orchard to timely discover possible pest infections. Orchards and vineyards are formed by several consecutive and adjacent aisles, which in turn contain many trees along a straight line. The nets on top of modern orchards (see Figure 1), originally set to protect the crops from hail, can make it more difficult for pests to enter and infest the trees [10], [11]. However, the nets constrict the drone to fly not over the trees, but only along the aisles.

Fig. 1. Modern orchards use nets on top (left) and on the sides of the trees (right) to protect crops from pests and bad weather.

For detecting bugs in modern orchards, a drone, moving along aisles, is more effective than a ground or handheld device because it can cover the whole area in a shorter time. Moreover, just by changing its height, a drone can take video/photos at the bottom or the top of each tree with the same focal distance, thus preserving the same spatial resolution. However, due to battery limitations, drones may not be able to cover the entire orchard. The drone's energy consumption does not depend only on the orchard size, but also on the payload (e.g., an RGB sensor is lighter than multi- or hyper-spectral sensors, but with limited spectral information) and on the speed. When recording video or taking photos, the drone must fly slowly (or even stop, i.e., hovering) to preserve adequate camera stabilization. Limited speeds mean longer coverage times and also, depending on the characteristics of the drone, higher energy consumption.

This work was partially supported by NSF grants CNS-1818942, OAC-1725755, OAC-2104078, and SCC-1952045; partially supported by the "GNCS – INdAM"; and also partially supported by "HALY-ID" project funded by the European Union's Horizon 2020 under grant agreement ICT-AGRI-FOOD no. 862665, no. 862671, and from MIPAAF.

2325-2944/22/$31.00 ©2022 IEEE
DOI 10.1109/DCOSS54816.2022.00032

Given that the drone has limited energy for flying and image/video recording, in this paper we assume that the observable positions on the trees can be prioritized. Specifically, we associate a priority, called *reward*, to each observable position. The reward can be decided by analyzing the historical time series of bug numbers, along with weather conditions that have been recorded in past years, and also by considering entomological knowledge. So, our objective is to plan a drone's route in the orchard for taking pictures or videos (to scout bugs) in the most profitable locations (with larger reward) under a given limited drone's battery. This problem can be modeled as an instance of the well-known Orienteering Problem (OP).

The contributions of this paper are summarized as follows.

- We define a novel optimization problem, called *Single-drone Orienteering Aisle-graph Problem* (SOAP[1]).
- We design three algorithms for solving SOAP, i.e., an optimal algorithm (OPT), and two fast heuristic algorithms (GBT and GBA).
- We evaluate the performance of our algorithms on randomly generated synthetic data and real data adapted to our problem. We also propose a real test-bed on an orchard for detecting bugs.

The rest of the paper is organized as follows. Section II reviews the related work. Section III formally defines the problem SOAP. Section IV presents algorithms for solving SOAP. Section V evaluates the effectiveness of our algorithms, and Section VI offers conclusions and future research directions.

II. Related Work

The OP is a particular variant of the classical Traveling Salesman Problem (TSP) problem where, in addition to the cost paid to move from one city to another, profits/rewards are added to the cities. Precisely, the goal of TSP is to find the minimum cost cycle in the area such that all cities are visited exactly once, while OP has to find a cycle among a subset of the cities that maximizes the sum of the profits of the visited cities without exceeding a given cost budget.

The OP in aisle-graphs was first studied in [13]. The authors studied the case of aisle-graphs with two accesses (two junction lines, one on each endpoint of the aisles) in which a robot has to plan a route inside the orchard for regulating the quantity of water to give to the trees in order to guarantee the proper moisture level. For solving this problem, the authors proposed two greedy heuristics, GFR, and GPR, which select a subset of full or partial aisles to traverse, respectively. The robot calculates the budget needed to collect the rewards from its current position and prefers full (respectively, partial) aisles with maximum reward per budget unit in GFR (respectively, GPR). The time complexities of GFR and GPR are $\mathcal{O}(m^2)$ and $\mathcal{O}(m^2 n)$, respectively, where m is the number of aisles and n is the number of vertices (i.e., trees) composing each aisle. In [14], the authors studied the problem of routing multiple robots, proposing three algorithms combining GFR and GPR both in series or in parallel.

The authors in [15] developed polynomial-time algorithms to improve some of the problems faced in [13] along with a rich set of experimental results on synthetic and real data set [16]. In particular, they designed an optimal algorithm, called OFR-I, that improves on GFR by determining the optimal solution for the full-row policy, whose time complexity is $\mathcal{O}(m \cdot \max\{n, \log m\})$. Moreover, they proposed HGC, which slightly improves on GPR at a higher time complexity. The work in [17] presents approximation and greedy algorithms for solving the OP problem in aisle-graphs with a single access. In [18] the authors consider the NP-hard stochastic OP, where the goal is to navigate between start and end vertices in a graph, maximizing the sum of rewards for visited vertices (i.e., trees) while obeying a travel budget over edges with a stochastic cost within a given probability of failure.

A further line of research is proposed in [19] where a symbiotic cooperation between ground robots and flying drones in aisle-graphs is investigated. The team of robots has to perform regular tasks like monitoring or collecting data in a precision agriculture scenario. Since robots are battery-powered vehicles, their autonomy in terms of life-time is limited, so mobile charging stations (drones and other kinds of robots) are employed for refueling the ground robots by swapping their batteries with new fresh ones. The authors, therefore, present the Refuel Scheduling Problem (RSP), whose objective is to plan a suitable scheduling for the team of mobile charging stations dispatched to replace the batteries on-board the ground robots, in order to guarantee a minimum service rate for the ground robots avoiding interruptions.

III. Problem Definition

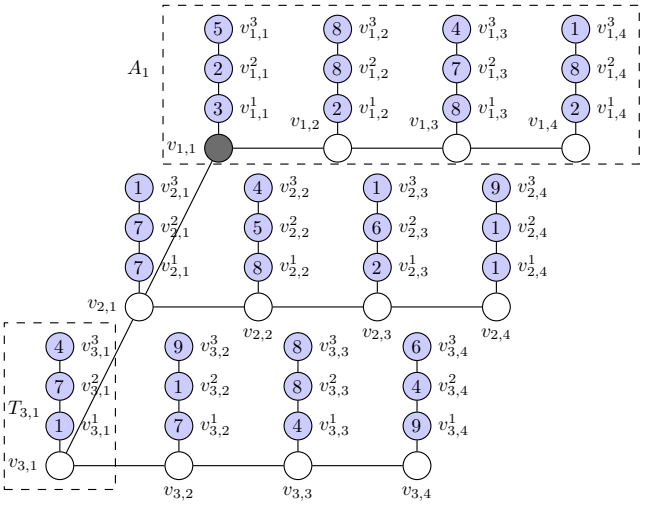
In this section we formally present the orchard graph model, the reward and cost functions, and the problem to solve.

A. Orchard Graph Model

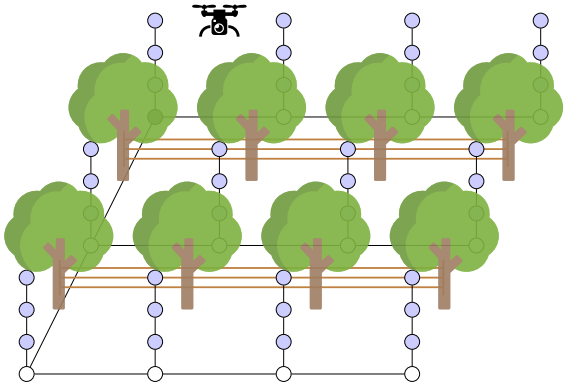
Let $O(m, n, l) = (V, E)$ be the *3-D single-access aisle-graph* (briefly, *orchard*) that represents the constrained structure formed by a set of vertices V and a set of edges E. Specifically, given the orchard $O(m, n, l)$, m denotes the number of *rows* (aisles), n denotes the number of *columns*, and l is the number of possible *observable positions* on each tree. However, rows are all connected only via the first column (also called *backbone*) (see Figure 2a). Since on each row there are n trees, therefore the orchard has exactly mn trees and mnl different positions to be observed by the drone.

Formally, the set of vertices is defined as $V = V^T \cup V^P$ where V^T denotes the actual *set of tree roots* of the orchard and V^P denotes the *set of observable positions* by the drone. In particular, $V^T = \{v_{i,j} | 1 \leq i \leq m, 1 \leq j \leq n\}$ while $V^P = \{v^k_{i,j} | 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq l\}$. Analogously, the set of edges is defined as $E = E^T \cup E^P$ where E^T denotes the *inter-tree connections*, while E^P denotes the *intra-tree connections*. In particular, E^T is defined as follows: each vertex $v_{i,j}$ with $1 \leq i \leq m$ and $1 < j < n$ has two edges,

[1]Insecticidal soap is used to control many plant insect pests [12].



(a) Orchard model. In particular, the aisle A_1 and the tree $T_{1,3}$ are highlighted. The rewards are inside the light-blue vertices, while the costs are omitted because they are assumed to be unitary.



(b) Sketched representation.

Fig. 2. An orchard $O(3,4,3)$ shown in two different representations.

one toward $v_{i,j-1}$ and the other toward $v_{i,j+1}$; each vertex $v_{i,1}$ with $1 < i < m$ has three edges: one toward $v_{i-1,1}$, one toward $v_{i+1,1}$, and one toward $v_{i,2}$. Accordingly, edges connected to the corner vertices $v_{1,1}$ are $v_{m,1}$ are well defined.

Concerning E^P, each vertex $v^k_{i,j}$ for any i, j with $1 \leq k < l$ has two edges, one toward $v^{k-1}_{i,j}$ and the other toward $v^{k+1}_{i,j}$ forming a connected list. By convention, let $v^0_{i,j} = v_{i,j}$. Accordingly, each tree located in $v_{i,j}$ has an edge to its associated list. Let $r_i = \{v_{i,1}, \ldots, v_{i,n}\}$ be the i^{th} row in the orchard with $i \leq 1 \leq m$, and let $c_1 = \{v_{1,1}, \ldots, v_{m,1}\}$ be the *backbone*, i.e., the only column in the orchard which connects all the rows together. Let A_i be the i^{th} *2-D single-access aisle-graph* (briefly, *aisle*) of $O(m,n,l)$ with n rows and l columns (here, the vertices on columns actually represent the observable positions), with $1 \leq i \leq m$ (see A_1 in Figure 2a). In other words, $O(m,n,l)$ consists of m 2-D single-access aisle-graphs, one for each aisle. Let $T_{i,j}$ be the *tree* in the orchard rooted at $v_{i,j}$ and vertices $v^1_{i,j}, \ldots, v^l_{i,j}$, with $1 \leq i \leq m, 1 \leq j \leq n$ (see $T_{3,1}$ in Figure 2a). Finally, the main vertex is the *depot* located at $v_{1,1}$.

Figure 2a depicts an example of an orchard $O(3,4,3)$ with 3 rows (aisles), 4 columns (number of trees for aisle), and 3 possible observable positions for each tree. Hence, there are 12 trees with overall 36 observable positions. The main vertex (depot) $v_{1,1}$ is depicted in dark gray representing the initial and final location of the drone. The trees are rooted on the white (and dark gray) vertices, while the observable positions are the light blue ones. Graphically, the same orchard $O(3,4,3)$ can be seen in Figure 2b. It is worthy to point out that the drone cannot cross the aisles because of the tree's foliage and cannot fly over because of the net.

B. Reward and Cost Functions

In the orchard, a drone is in charge of visiting a subset of trees for doing specific tasks like videos or pictures. The drone is an energy-constrained flying vehicle that has a battery of capacity $B \geq 0$ in terms of energy, and when the drone moves flying along edges, it spends energy from its battery. Let $\mathcal{C} : E \to \mathbb{R}^+$ be the *edge cost function* that defines the energy cost for flying along any edge $e \in E$. For simplicity, we consider *constant* the edge cost function, i.e., $\mathcal{C}(e) = \gamma$ for each $e \in E$, for a given $\gamma \in \mathbb{N}$. Moreover, without loss of generality, we assume unitary costs, i.e., $\gamma = 1$. This assumption can be considered counter-intuitive because an edge between two adjacent trees is much longer than that between two consecutive observable positions on a tree. Nevertheless, the time required for the drone to completely move between two adjacent trees, or two adjacent locations on a tree, can be reasonably considered the same due to accelerations and decelerations that the drone needs to perform when moving. Moreover, when the drone takes pictures, it needs to stay in place for some time to stabilize the camera and get proper focus. Hence, for ease of simplicity, the energy required for moving can be assumed to be the same even if the distances are different.

Let $\mathcal{R} : V \to \mathbb{R}^+$ be the *reward function* that characterizes the importance of doing a specific task on vertices. The reward function provides meaningful values only for the set of observable positions V^P while giving zero values for the set of trees, hence $\mathcal{R}(v) = 0$ for any $v \in V^T$. The reward associated with each vertex is assigned to the drone only the first time the drone performs tasks on that particular vertex. Further visits at the same vertex do not collect any reward. However, if the drone repeatedly travels on the same edge multiple times, the energy cost is considered multiple times. Given a subset of vertices $X \subseteq V$, there is a unique minimum cycle trip to/from the depot vertex that visits all the vertices in X because the structure is loosely connected. The graph induced by the vertices in X is a tree rooted at $v_{1,1}$. For a given $X \subseteq V$, let $\overline{\mathcal{C}}(X)$ be the minimum *overall flying cost* for the drone defined as the sum of the costs on edges for the unique path for connecting all the vertices in X. Similarly, let $\overline{\mathcal{R}}(X)$ be the *overall attainable reward* defined as the sum of the individual rewards of each vertex in X (plus others that are in the unique cycle induced by the vertices in X).

C. Problem Formulation

In this paper, we consider a drone that has to take photos or record videos inside an orchard to scout bugs. Specifically, the drone plans a route inside the orchard to/from the main depot while considering its available battery capacity. Given an orchard $O(m, n, l)$ with m rows, n columns, and l observable positions on each tree, and given a battery capacity $B \geq 0$, the aim of the *Single-drone Orienteering Aisle-graph Problem* (SOAP) is to find a suitable cycle for the drone that starts and finishes at the depot $v_{1,1}$ such that the collected reward is maximized and its energy cost is no larger than B. Indeed, SOAP is a special case of the OP problem. Given any subset X of vertices in O, SOAP aims to select a subset S such that maximizes the reward under the constraint that the traveling cost to connect all the locations in S does not overtake a given budget B. Formally:

$$S = \arg\max_{X \subseteq V} \{\overline{\mathcal{R}}(X) \mid \overline{\mathcal{C}}(X) \leq B\}, \tag{1}$$

where $\overline{\mathcal{R}}$ is the reward function to maximize, $\overline{\mathcal{C}}$ is the cost function, and B is the budget given in input.

Recalling that the costs are constant and unitary in SOAP, the minimum budget in input must be at least 2 ($B_{\text{min}} = 2$). This is due to the fact that the closest vertex with potential reward from the depot is $v^1_{1,1}$, which is adjacent to it. Moreover, $B = 2$ guarantees a non-trivial round trip in the orchard O. On the other hand, the budget B is upper-bounded and cannot be larger than the budget that ensures a sequential and full visit of the orchard, i.e., $B_{\text{max}} = m2(n-1)+2(m-1)+mn2l$, where the first term is the cost for traveling (back and forth) all the rows, the second term is the cost for traveling the backbone, and the third term is the cost for visiting all the trees in O. Any budget larger than B_{max} would result in a waste of energy. So, a meaningful value of budget is $B_{\text{min}} \leq B \leq B_{\text{max}}$, and asymptotically it holds that $B = \mathcal{O}(mnl)$.

IV. Proposed Algorithms for SOAP

In this section we propose a polynomial-time optimal algorithm, called OPT, and two time-efficient heuristic greedy algorithms, called Greedy Best Tree (GBT) and Greedy Best Aisle (GBA).

A. The Optimal OPT Solution

We devise a dynamic programming algorithm, called OPT, that optimally solves SOAP. It makes use of 4 tables.

The first table $\mathbb{T}$ stores, for each tree $T_{i,j}$ with $1 \leq i \leq m, 1 \leq j \leq n$, the cumulative reward that can be attained given a budget $2k$ such that $0 \leq k \leq l$ from the initial vertex $v_{i,j}$. Clearly, $\mathbb{T}[i, j, 0] = 0$ by convention. Hence, we can fill the table as follows: $\mathbb{T}[i, j, k] = \mathbb{T}[i, j, k-1] + \mathcal{R}(v^k_{i,j})$. The algorithm makes use of this table because it has to decide how many consecutive vertices will be part of the solution for each tree. The size of table $\mathbb{T}$ is $mn(l+1)$.

The second table $\mathbb{A}$ stores, for each aisle A_i with $1 \leq i \leq m$ and any budget $i - 1 \leq b \leq \lfloor \frac{B}{2} \rfloor$, the best solution so far computed. Indeed, for each aisle A_i the algorithm has to decide which trees to include in the solution and how much budget to dedicate to each tree. Since to take the tree $T_{i,j}$ one must traverse all the trees from 1 to $j-1$ in the aisle A_i, possibly without taking any vertex in some of those trees, we create a table $\mathbb{A}$ of size $m \times n \times (nl + (n-1))$ where:

$$\mathbb{A}[i, j, b] = \max_{0 \leq k \leq l} \{\mathbb{A}[i, j-1, b-k-1] + \mathbb{T}[i, j, k]\}. \tag{2}$$

Note that $\mathbb{A}[i, j, b]$ visits the aisle A_i up to the tree $T_{i,j}$, and when $k = 0$ no vertices of that tree are visited.

Then, in the third table $\mathbb{A}^*$, for each aisle A_i with $1 \leq i \leq m$ and any budget $i - 1 \leq b \leq \lfloor \frac{B}{2} \rfloor$, we memorize which is the maximum reward by varying the last visited tree:

$$\mathbb{A}^*[i, b] = \max_{1 \leq j \leq n} \{\mathbb{A}[i, j, b]\}. \tag{3}$$

The size of table $\mathbb{A}^*$ is $n \times (nl + (n-1))$.

Finally, OPT has to decide which aisle to select and for each aisle, up to which tree to visit. Considering that to get to the aisle A_i one must traverse the part of the backbone c_1 in front of the rows $1, \ldots, i-1$ up to row i, the optimal solution including A_i is defined by the optimal solution that includes up to A_{i-1}, possibly without taking any tree in some of the aisle $A_1, \ldots, A_i$. For this, we create the fourth table $\mathbb{O}$. To find the final solution, first we compute the maximum reward with a given budget b assuming to traverse up to row i.

$$\mathbb{O}[i, b] = \max_{0 \leq b' \leq \min\{b-i-1, nl+n-1\}} \{\mathbb{O}[i-1, b-1-b'] + \mathbb{A}^*_i[b']\}. \tag{4}$$

Then, among $\mathbb{O}[i, b]$, we compute the absolute maximum reward with budget bounded by B as:

$$\max_{1 \leq i \leq m, 0 \leq b \leq \lfloor \frac{B}{2} \rfloor} \{\mathbb{O}[i, b]\}. \tag{5}$$

Note that the size of table $\mathbb{O}$ is $n \times \lfloor B/2 \rfloor$.

The algorithm runs in time $\mathcal{O}(mnlB)$ time. Since the maximum budget is B_{max}, i.e., B is upper bounded by $\mathcal{O}(mnl)$, then algorithm OPT has time complexity $\mathcal{O}(m^2n^2l^2)$.

By the above discussion:

Theorem 1. *Algorithm* OPT *optimally solves* SOAP.

Proof. Omitted due to page limit constraints. □

Let us illustrate how to compute $\mathbb{O}$, i.e., the result of OPT, with the example in Figure 2a considering $B = 21$, and assuming to have already computed by Eq. (3) the values $\mathbb{A}^*[i, b]$, for $0 \leq i \leq 2$ and $0 \leq b \leq 10$, reported in Table I.

TABLE I
Table $\mathbb{A}^*$ considering the example in Figure 2a with $B = 21$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	3	5	10	18	21	26	33	37	40	43
2	0	7	14	15	22	27	31	32	35	39	40
3	0	1	8	12	17	20	27	29	37	40	46

Let us consider the best solutions when taking into account only the first aisle A_1. $\mathbb{O}[1, 1] = 3$ because in the first aisle and budget 2 (recall that the second index in $\mathbb{O}$ is halved,

so 1 means budget 2) only the vertex $v^1_{1,1}$ can be visited, whose reward is 3. $\mathbb{O}[1,2] = 5$ because the drone can visit both $v^1_{1,1}$ and $v^2_{1,1}$. It is interesting to see that $\mathbb{O}[1,3] = 10$ has two possible choices, i.e., either completely visiting the first tree $T_{1,1}$ ($3+2+5=10$) or visiting the vertices $v^1_{1,2}$ and $v^2_{1,2}$ from the second tree ($2+8=10$). With the same reasoning, $\mathbb{O}[1,4] = 18$ up to the last value $\mathbb{O}[1,10] = 43$, which is given by $v^1_{1,2}, v^2_{1,2}, v^3_{1,2}, v^1_{1,3}, v^2_{1,3}, v^2_{1,4}, v^3_{1,4}$. That is, $\mathbb{O}[1,b] = \mathbb{A}[1,b]$, for any b.

Let us consider the best solutions when considering also the second aisle A_2. Starting from this phase, we have to redistribute the budget to the new current aisle and to the previous ones. For computing $\mathbb{O}[2,1]$ we have to compare the best solution when assigning budget 2 for A_2 (and 0 for A_1), and the best solution when giving budget 0 for A_2 (and 2 for A_1). In this case, $\mathbb{O}[2,1] = \max\{0,3\}$ because with budget 2 we cannot visit any observable position in A_2 (this is the reason why we have 0) while the best solution assigning budget 2 up to the previous aisle was 3: hence the maximum is 3. For $\mathbb{O}[2,2]$ we have an additional comparison. In fact, we can distribute the budget as follows: 4 only for A_2, 2 for A_1, or 4 for A_1. Hence, $\mathbb{O}[2,2] = \max\{7,3,5\}$ because in the first case, we can visit $v^1_{2,1}$ whose reward is 7, in the second case we cannot obtain any reward from A_2 but the best solution so far calculated up to A_1 with budget 2 gives 3, and in the third case 0 reward from A_2 and the best solution with budget 4 was with reward 5. So, the maximum value among those is 7. With the same reasoning, $\mathbb{O}[2,3] = \max\{14,10,3,5,10\} = 14$ up to the last value $\mathbb{O}[2,10] = 47$ obtained assigning budget 4 to A_2 and budget 7 to A_1.

The same logic is applied when considering A_3. In fact, for example in $\mathbb{O}[3,2]$ we compare the best solutions assigning budget 4 for only A_3, or budget 2 for A_3 and the best solution with budget 2 up to A_2, or the best solution with budget 4 up to A_2. Hence, $\mathbb{O}[3,2] = \max\{0,3,7\}$ because in the first case we cannot visit any observable position in A_3, in the second case, we already know that with a budget of 2 the best solution gives 3, while in the third case the best solution is 7. Hence, the maximum value is 7. The last value for A_3 is $\mathbb{O}[3,10] = 47$.

TABLE II
TABLE $\mathbb{O}$ CONSIDERING THE EXAMPLE IN FIGURE 2A WITH $B = 21$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	3	5	10	18	21	26	33	37	40	43
2	0	3	7	14	18	22	27	33	37	40	**47**
3	0	3	7	14	18	22	27	33	37	40	**47**

Finally, the total reward is 47 with $B = 20$ (see Table II): budget 14 is spent for A_1, and budget 4 for A_2 (see Table I).

B. The GBT *Heuristic Algorithm*

In this section, we devise a faster algorithm that sub-optimally solves SOAP in polynomial time, called GREEDY BEST TREE (GBT).

When a tree is selected in this strategy, all the observable positions are visited. So, we say that the main idea behind the strategy is to consider only fully visited trees. At each iteration, the tree reachable with the current budget with the largest ratio reward/cost is selected. The algorithm finishes when either $B = 0$ or the budget is not enough to reach any unselected tree.

The pseudo-code of GBT is given in Algorithm 1.

Algorithm 1: The GBT Algorithm

```
1 Ŝ ← ∅
2 while B > 0 and T_{i,j} ≠ ∅ do
3     T_{i,j} ← arg max R̄(T_{i,j}) / C̄(T_{i,j}) s.t.
4                 T_{i,j} ∉ Ŝ and B − C̄(T_{i,j}) ≥ 0
5     Ŝ ← Ŝ ∪ T_{i,j}, B ← B − C̄(T_{i,j})
6 return Ŝ
```

Initially the solution $\hat{S}$ is empty (Algorithm 1, Line 1), and the main cycle starts (Line 2). The drone will start to consider only a subset of trees visited as a whole. This means that once the drone begins to visit the first observable position $v^1_{i,j}$ on a particular tree $T_{i,j}$, it has necessarily to continue until the last observable position $v^l_{i,j}$. Therefore, for each tree $T_{i,j}$ in the orchard, the GBT initially sums up all the possible obtainable rewards and the costs (for visiting the whole tree plus the traveling cost for reaching that tree given the current solution). In Line 4 we compute $\overline{\mathcal{R}}(T_{i,j})$ which is the sum of the rewards of vertices for the tree $T_{i,j}$, and $\overline{\mathcal{C}}(T_{i,j})$ which is the total cost for completely visiting the same tree also considering the (possible additional) traveling cost for reaching that tree given the current solution. After that, we greedily pick the tree $T_{i,j}$ that has the largest ratio between the sums of the rewards and the sums of the costs (Line 4). Then, both the current solution and budget are updated (Line 5). This selection is repeatedly performed until the budget is completely used. If there is no available tree to be picked, the cycle stops.

About the time complexity, since the number of trees $T_{i,j}$ is mn, the main cycle is repeated for $\mathcal{O}(mn)$ times. Every time we evaluate all the ratios among the unpicked trees. So the time complexity of GBT is $\mathcal{O}(m^2n^2)$.

Let us illustrate the GBT algorithm with the example in Figure 2a. We consider $B = 21$. Iteratively, we consider the rewards and costs of each tree. The first tree $T_{1,1}$ has a total reward 10 and cost 6, and hence ratio 1.67. Analogously, the second tree $T_{1,2}$ has a total reward of 18 and cost of 8, and hence a ratio of 2.25, and so on. When evaluating all the trees, the tree with the largest ratio is $T_{1,2}$, and so it is added to the solution. After the first selection, we reconsider the remaining trees. For $T_{1,1}$ we have again the same ratio 1.67. The tree $T_{1,3}$ has reward 19 and now cost 8 (ratio of 2.375), because we already considered the cost for reaching $T_{1,2}$, and so on. Now, the tree with the largest ratio is $T_{1,3}$, and it is selected.

Finally, this algorithm returns 37 as the total reward and 16 as the cost. Notice that it remains a $21 - 16 = 5$ residual budget which is not sufficient for visiting any other tree as a whole.

C. The GBA Heuristic Algorithm

In this section, we devise another heuristic algorithm that solves SOAP in polynomial time, called GREEDY BEST AISLE (GBA). This algorithm either visits an entire aisle or discards it. Therefore, the main idea behind the strategy is to consider only fully visited aisles in the solution. At each selection, the aisle with the maximum ratio between reward and cost is preferred. The pseudo-code of GBA is given in Algorithm 2.

Algorithm 2: The GBA Algorithm

```
1 Ŝ ← ∅
2 while B > 0 and A_i ≠ ∅ do
3 |   A_i ← arg max R̄(A_i)/C̄(A_i) s.t.
4 |           A_i ∉ Ŝ and B − C̄(A_i) ≥ 0
5 |   Ŝ ← Ŝ ∪ A_i, B ← B − C̄(A_i)
6 return Ŝ
```

Initially the solution $\hat{S}$ is empty (Algorithm 2, Line 1), and the main cycle starts (Line 2). The drone will start to consider only a subset of aisles visited as a whole. Basically, it follows the same reasoning as done for GBT, with the only difference that instead of selecting full trees $T_{i,j}$ here we select full aisles A_i. The most convenient aisle A_i, in terms of ratio reward/cost $\overline{\mathcal{R}}(A_i)/\overline{\mathcal{C}}(A_i)$, is greedily selected (Line 4). As usual, when considering the cost, we also consider the possible additional traveling cost (in the backbone) for reaching the picked aisle A_i. About the time complexity, GBA requires to precompute the sum of rewards and costs only once, requiring $\mathcal{O}(mnl)$ in time. Then, we pick the best aisles until the budget is finished.

It is worthy to note that GBA can end up with a significant amount of budget, although not enough to visit an entire aisle. Thus, the algorithm can extend the solution by exhausting the budget, e.g., by selecting a portion of the aisle that would be chosen by the greedy strategy. This last step is not reported in GBA. The computational cost of the extended solution for GBA does not exceed the total computational of the original algorithm, which remains $\mathcal{O}(mnl)$.

Let us illustrate the GBA algorithm with the example in Figure 2a. We consider $B = 21$. For the whole aisle A_1, the total reward is 58 with cost 30 and ratio 1.93. For A_2, the total reward is 52 with cost 32 and ratio 1.73, and for A_3, the total reward is 68 with cost 34 and ratio 2.27. So, the most convenient aisle is A_3. However, its cost of 34 is larger than the available budget of 21. So, GBA will end up with the entire budget $B = 34$ unused. To extend the solution, we apply GBT to the aisle A_3 with a budget of 21, left after considering the backbone cost to reach A_3. The solution achieves a total reward of 37 and ends up with 1 unit of budget unused.

V. PERFORMANCE EVALUATION

We implemented our algorithms in Python language version 3.7, and run all the instances on an Intel i7-10genK computer with 16 GB of RAM. We evaluate the performance, in terms of total obtained reward, of the presented algorithms for solving SOAP, i.e., OPT, GBT, and GBA.

We tested the algorithms on three different workloads W_i:

- W_1: randomly generated synthetic data sets;
- W_2: real data set from an orchard test-bed for insect scouting; and
- W_3: a large data set from an irrigation application for smart agriculture.

For W_1, we run different scenarios varying the number of rows m, the number of columns n and the number of observable locations l, and the distribution of the rewards by changing appropriate parameters. For each scenario, each algorithm is tested with an increasing budget $B = \{5\%, 10\%, 15\%, 20\%, 40\%, 60\%, 80\%\}B_{\max}$ (x-axis), and we plot the average of the results on 33 instances along with their 95% confidence interval (y-axis). A few words about the y-axis: here, we plot the *ratio* among the total reward collected by the tested algorithm and the one collected by OPT. Obviously, such a ratio should be less than or equal to 1. For W_2, we implemented our algorithms on a drone and created a demonstrative test-bed in which we provide as input rewards set manually from previous knowledge of the orchard. Finally, for W_3, we have 10 instances from an irrigation application for smart agriculture that is opportunely adapted for SOAP.

A. With Synthetic Data set

In the workload W_1, we randomly generated rewards on vertices through the ZipF distribution [20] by varying the θ parameter. By varying the θ parameter we can study how much the variability in the orchard can affect our algorithms. Note that in our evaluation we assume to have the rewards (integers) in the interval $[0, 100)$. So, when $\theta = 0$, the rewards are uniformly distributed in $[0, 100)$, while when θ increases, the smallest rewards become more and more frequent than the largest ones. In particular, in this workload we assume $\theta = \{0, 0.8\}$. Moreover, we vary the layout of the orchard O by setting the number of rows $n = \{50, 25\}$, the number of columns $m = \{25, 50\}$, and fixing the number of observable locations $l = \{3, 5\}$. Figure 3 comprehensively illustrates the evaluation on synthetic data sets.

We first observe the impact of θ. As we can see, the plots on the left have $\theta = 0$ while the others on the right have $\theta = 0.8$. Although all the greedy algorithms work almost the same, with $\theta = 0.8$ there is a gap of about 10% in favor of the case with $\theta = 0$. This shows that the greedy heuristics, which are confined to select large portions (full trees or full rows) waste budget because the reward is very variable and it does not satisfy any locality rule.

Comparing the two heuristics GBT and GBA, we can see that when the budget is low, GBT obtains better performance than GBA, while when the budget in input increases, the behavior is exactly the opposite, with GBA better than GBT. This is due to the different granularity of the choice of two algorithms: when the budget is limited, still GBT can select the most advantageous tree with respect to the actual budget,

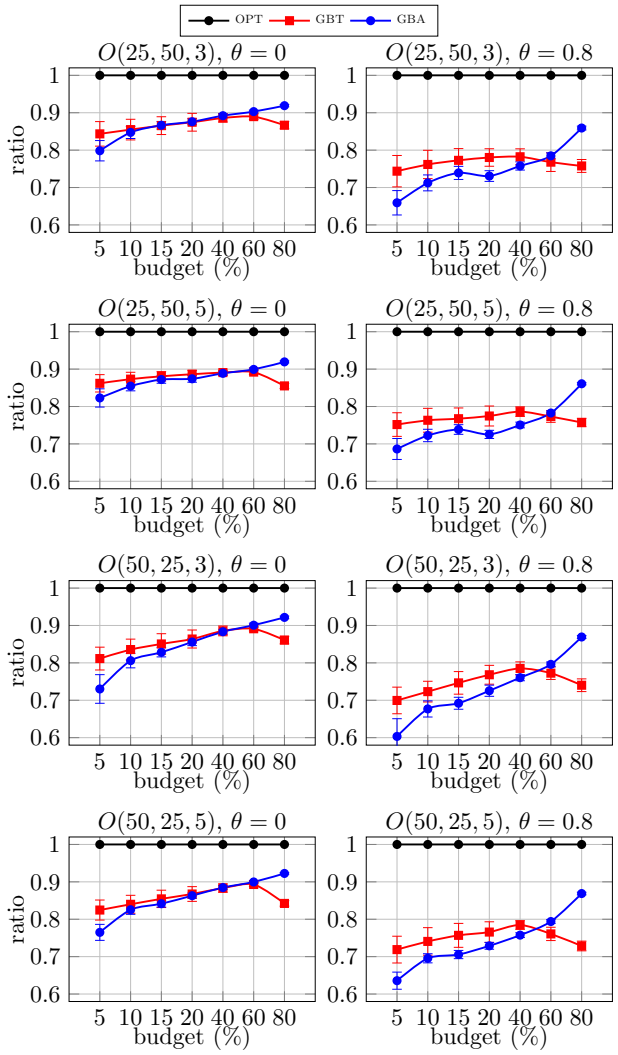


Fig. 3. W_1: randomly generated synthetic data sets.

while GBA feels much more the budget constraint and it is forced to select a closer aisle independently of its reward. Vice versa, when the budget is large, GBA, which selects full aisles, is less myopic than GBT. In fact, GBT may end up trapped in local minima because it prefers sparse trees that, although have a higher reward, lead to a loss of budget.

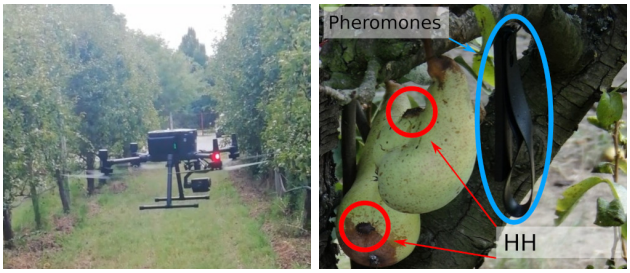
By varying the number l of observable positions (respectively, the number n of trees), we see that the performance of the algorithms slightly differs. In particular, GBT (respectively, GBA) obtains more reward when l (respectively, n) increases.

Finally, we have compared two different layouts, i.e., more columns than rows ($m < n$) and more rows than columns ($m > n$). The results highlight that in the previous case ($m = 25, n = 50$) the performance is in general slightly better than the latter case (around 5%). This is probably influenced by the energy required for traveling in the backbone.

B. Preliminary Experimental Results

In order to test the effectiveness of our algorithms in a real-world scenario, we arranged a small real test-bed using a drone, creating the workload W_2. This experiment has been performed in the context of the HALY-ID research project [21] that aims to scout the *Halyomorpha Halys* (HH) bug (shown in Figure 4b). HH seriously damages fruit production (pears in particular) not only in northern Italy but also in central Europe, the United States, and eastern Asia. In our small test-bed, we aimed at scouting the presence on the trees of HH attracted by previously placed pheromones (shown in Figure 4b).

The orchard of our test-bed has 12 rows and 15 columns (180 pear trees), and we set $t = 3$ observable positions, having so an $O(12, 15, 3)$. About the drone, we used a powerful DJI Matrice 300 RTK (see Figure 4a) which is able to fly for up to $55\,\text{min}$ having 2 batteries with capacity of $\approx 6000\,\text{mAh}$ [22].



(a) (b)

Fig. 4. Our developed test-bed. a) the drone used; b) original footage, showing the Halyomorpha Halys (HH) captured by the drone.

Since only a small selection of trees had the pheromones, we modeled this instance by adding more reward to the trees closer to the pheromones, and less reward otherwise. So, the objective was to take pictures of the trees in order to collect as many meaningful pictures with the HH as possible. On each tree, we set 3 observable positions: bottom, middle, and top. The redistribution of the reward in each tree ensures that the observable positions on the top of the trees have more reward, as per entomological knowledge. Precisely, we redistribute the reward in each tree $v_{i,j}$ with pheromones assigning 20%, 30%, and 50% of the reward to $v^1_{i,j}$, $v^2_{i,j}$, and $v^3_{i,j}$, respectively.

Figure 5 (left) depicts the results of the proposed algorithms. We observe that in this real data set the variability is particularly high and this is given by how the pheromones were positioned. Since some of the pheromones have been placed close to the depot, it can be seen that with a very low budget (5%) most of the algorithms return solutions very close to the optimal. As the budget increases, the total reward increases for each algorithm, but the totalized ratio actually decreases until budget approximately $\geq 40\%$. GBT and GBA perform differently, with GBT more stable, in terms of obtained ratio than GBA. This depends on the point-wise reward associated with trees: the reward of an aisle is much less predictable in presence of pheromones than that of a tree. However, in general, with this small test-bed, all the algorithms perform very well, with $\geq 80\%$ of the optimum reward given by OPT.

C. With Real Data set

In the workload W_3, we use a real data set that comes from another smart agriculture application [16] in order to test our algorithms on larger orchards. The field is composed of 137×107 trees. The rewards indicate how important is to regulate the quantity of water to the trees. However, in that data set, we only have a single reward for each tree. To adapt that data set to our application, as before we set 3 observable positions on each tree: the bottom, the middle, and the top, and we redistribute the reward as before. Since the corresponding graph is quite large, i.e., $O(137, 107, 3)$, we only run the two heuristics GBT and GBA. Figure 5 (right) shows the results.

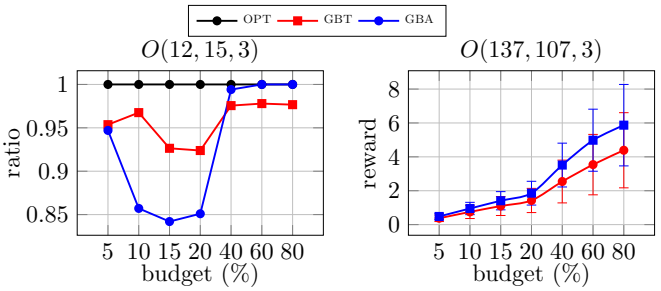


Fig. 5. W_2 (left) and W_3 (right).

First of all, we cannot report in the y-axis the *ratio* since we did not run the OPT. Hence, we only show the retrieved reward scaled down by a factor of 10^4. The peculiarity of W_3 is that the need for water (i.e., reward) is split into macro areas that include multiple trees. Moreover, in each of such areas, the reward smoothly increases or decreases. Therefore, as discussed earlier, GBA can be considered the winning strategy in this context, also given its lower time complexity.

VI. Discussion and Outlook

Drones are revolutionizing agriculture by offering farmers efficacy and efficiency, and early pest scouting is a major application of drones in this area. The impact of drones is constrained by the energy required for flying and collecting data. Thus, an important step towards bug detection is to plan an efficient route for the drone in the orchard by selecting the areas/trees in which to capture images or videos. In this paper we studied this scenario by introducing a new problem, called SOAP, and a graph model to represent real orchards. We optimally solved SOAP, devising also two heuristic algorithms with a reduced computational time cost.

In future work, we would like to integrate our algorithm with a machine learning approach to perform comprehensive simulations and study its effectiveness and accuracy in real orchards. Moreover, it would be interesting to investigate a stochastic reward that can also account for the bug mobility. Also, a model with different costs for different edges is worthy to be explored to make our model more realistic. Symbiotic systems where multiple drones and ground devices collaborate have to be devised in next future.

References

[1] A. Khanda, F. Corò, F. B. Sorbelli, C. M. Pinotti, and S. K. Das, "Efficient route selection for drone-based delivery under time-varying dynamics," in *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, pp. 437–445, IEEE, 2021.

[2] F. B. Sorbelli, F. Corò, S. K. Das, L. Palazzetti, and C. M. Pinotti, "Greedy algorithms for scheduling package delivery with multiple drones," in *23rd International Conference on Distributed Computing and Networking (ICDCN)*, 2022.

[3] T. Calamoneri, F. Corò, and S. Mancini, "A realistic model to support rescue operations after an earthquake via uavs," *IEEE Access*, 2022.

[4] C. Qu, R. Singh, A. E. Morel, F. B. Sorbelli, P. Calyam, and S. K. Das, "Obstacle-aware and energy-efficient multi-drone coordination and networking for disaster response," in *2021 17th International Conference on Network and Service Management (CNSM)*, pp. 1–9, IEEE, 2021.

[5] A. Rabello, R. C. Brito, F. Favarim, A. Weitzenfeld, and E. Todt, "Mobile system for optimized planning to drone flight applied to the precision agriculture," in *2020 3rd International Conference on Information and Computer Technologies (ICICT)*, pp. 12–16, IEEE, 2020.

[6] F. H. Iost Filho, W. B. Heldens, Z. Kong, and E. S. de Lange, "Drones: innovative technology for use in precision pest management," *Journal of economic entomology*, vol. 113, no. 1, pp. 1–25, 2020.

[7] S. K. Dara, "The new integrated pest management paradigm for the modern age," *Journal of Integrated Pest Management*, vol. 10, no. 1, p. 12, 2019.

[8] R. Basri, F. Islam, S. B. Shorif, and M. S. Uddin, "Robots and drones in agriculture—a survey," in *Computer Vision and Machine Learning in Agriculture*, pp. 9–29, Springer, 2021.

[9] G. Sujayanand, S. Sheelamary, and G. Prabhu, "Recent innovations and approaches for insect pest management in agriculture," *Biotica Research Today*, vol. 3, no. 2, pp. 100–102, 2021.

[10] S. P. Giuseppino, B. P. Paolo, N. Roberta, M. Leonardo, and R. P. Federico, "Efficacy of long lasting insecticide nets in killing halyomorpha halys in pear orchards," *Outlooks on Pest Management*, vol. 29, no. 2, pp. 70–74, 2018.

[11] I. Schlathölter, A. Dalbosco, M. Meissle, A. Knauf, A. Dallemulle, B. Keller, J. Romeis, G. A. Broggini, and A. Patocchi, "Low outcrossing from an apple field trial protected with nets," *Agronomy*, vol. 11, no. 9, p. 1754, 2021.

[12] R. Weinzierl and T. Henn, "Botanical insecticides and insecticidal soaps," in *Handbook of integrated pest management for turf and ornamentals*, pp. 541–555, CRC Press, 2020.

[13] T. C. Thayer, S. Vougioukas, K. Goldberg, and S. Carpin, "Routing algorithms for robot assisted precision irrigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2221–2228, IEEE, 2018.

[14] T. C. Thayer, S. Vougioukas, K. Goldberg, and S. Carpin, "Multi-robot routing algorithms for robots operating in vineyards," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 14–21, IEEE, 2018.

[15] F. B. Sorbelli, S. Carpin, F. Corò, A. Navarra, and C. M. Pinotti, "Optimal routing schedules for robots operating in aisle-structures," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4927–4933, IEEE, 2020.

[16] F. B. Sorbelli, S. Carpin, F. Corò, S. K. Das, A. Navarra, and C. M. Pinotti, "Speeding up routing schedules on aisle graphs with single access," *IEEE Transactions on Robotics*, 2021.

[17] F. B. Sorbelli, F. Corò, S. K. Das, A. Navarra, and C. M. Pinotti, "Speeding-up routing schedules on aisle-graphs," in *2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 69–76, IEEE, 2020.

[18] T. C. Thayer and S. Carpin, "An adaptive method for the stochastic orienteering problem," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4185–4192, 2021.

[19] T. Gao, Y. Tian, and S. Bhattacharya, "Refuel scheduling for multirobot charging-on-demand," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5825–5830, IEEE, 2021.

[20] C. Tullo and J. Hurford, "Modelling zipfian distributions in language," in *Proceedings of language evolution and computation workshop/course at ESSLLI*, pp. 62–75, 2003.

[21] HALY.ID, "Project." https://www.haly-id.eu, 2022.

[22] DJI, "Matrice 300 rtk." https://www.dji.com/en/matrice-300/specs, 2022.